



Nível 3: BackEnd sem banco não tem
Ana Clara Quidute Marinho Silva - 202302798691

Campus Polo Setor Central - Araguaína(TO)

Desenvolvimento Full-Stack - Turma 2023.1 - Semestre Letivo 2024.1

Objetivo da prática:

1. Implementar persistência com base no middleware JDBC.
2. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
3. Implementar o mapeamento objeto-relacional em sistemas Java.
4. Criar sistemas cadastrais com persistência em banco relacional.
5. No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Criação do pacote `cadastrobd.model`:

```
- Classe Pessoa
/*
 * Click
 nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
 change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */
package cadastrobd.model;

/**
 *
 * @author anaqu
 */
public class Pessoa {
    private int idPessoa;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public Pessoa() {}
```

```
    public Pessoa(int idPessoa, String nome, String logradouro, String
cidade, String estado, String telefone, String email){
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

    public int getId_pessoa() {
        return idPessoa;
    }

    public void setId_pessoa(int idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return telefone;
    }
}
```

```

    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void exibir(){
        System.out.println("ID: " + idPessoa);
        System.out.println("Nome: " + nome);
        System.out.println("Logradouro: " + logradouro);
        System.out.println("Cidade: " + cidade);
        System.out.println("Estado: " + estado);
        System.out.println("Telefone: " + telefone);
        System.out.println("Email: " + email);
    }
}

```

- Classe PessoaFisica

```

/*
 * Click
 nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
 change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */
package cadastrobd.model;

/**
 *
 * @author anaqu
 */
public class PessoaFisica extends Pessoa{
    private String cpf;
    public PessoaFisica(){
        super();
    }
    public PessoaFisica(int idPessoa, String nome, String logradouro, String
cidade, String estado, String telefone, String email, String cpf){
        super();
        this.cpf = cpf;
    }

    public String getCpf() {
        return cpf;
    }
}

```

```

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
    @Override
    public void exhibir(){
        super.exibir();
        System.out.println("CPF: " + cpf);
    }
}

```

- Classe PessoaJuridica

```

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastrbd.model;

/**
 *
 * @author anaqu
 */
public class PessoaJuridica extends Pessoa{
    private String cnpj;

    public PessoaJuridica(){
        super();
    }
    public PessoaJuridica(int idPessoa, String nome, String logradouro,
String cidade, String estado, String telefone, String email, String cnpj){
        super();
        this.cnpj = cnpj;
    }

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
    @Override
    public void exhibir(){
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}

```

Criação do pacote **cadastro.model.util**:

- Classe ConectorBD

```
/*
 * Click
 nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
 change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */
package cadastro.model.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;

/**
 *
 * @author anaqu
 */
public class ConectorBD {

    private Connection c;
    private PreparedStatement ps;
    private ResultSet rs;

    private static final String URL =
"jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCe
rtificate=true";
    private static final String USUARIO = "loja";
    private static final String SENHA = "loja";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USUARIO, SENHA);
    }

    public PreparedStatement getPrepared(String sql) throws SQLException{
        c = getConnection();
        return c.prepareStatement(sql);
    }

    public ResultSet getSelect(String sql)throws SQLException{
        PreparedStatement ps = getPrepared(sql);
        rs = ps.executeQuery();
        return rs;
    }

    // Métodos close sobrecarregados

    public static void close(Connection c){
        if(c != null){
```

```

        try {
            c.close();
        } catch (SQLException excep){
            excep.printStackTrace();
        }
    }
}

public static void close (PreparedStatement ps){
    if(ps != null){
        try{
            ps.close();
        } catch(SQLException excep){
            excep.printStackTrace();
        }
    }
}

public static void close (ResultSet rs){
    if(rs != null){
        try{
            rs.close();
        } catch(SQLException excep){
            excep.printStackTrace();
        }
    }
}
}

```

- Classe SequenceManager

```

/*
 * Click
 nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
 change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */
package cadastro.model.util;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 *
 * @author anaqu
 */
public class SequenceManager {

    public SequenceManager() {}

```

```

    public static int getValue(String sequenceName){
        Connection c = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        int nextValue = -1;

        try{
            c = ConectorBD.getConnection();
            String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS
next_value";
            ps = c.prepareStatement(sql);

            rs = ps.executeQuery();

            if(rs.next()){
                nextValue = rs.getInt("next_value");
            }
        } catch (SQLException excep){
            excep.printStackTrace();
        } finally{
            ConectorBD.close(c);
            ConectorBD.close(ps);
            ConectorBD.close(rs);
        }
        return nextValue;
    }
}

```

Classes no padrão DAO, no pacote **cadastro.model**:

- Classe PessoaFisicaDAO

```

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastro.model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;

import cadastrobd.model.PessoaFisica;

```

```

/**
 *
 * @author anaqu
 */
public class PessoaFisicaDAO {

    public PessoaFisica getPessoa(int id){
        PessoaFisica pessoa = null;
        Connection c = null;
        PreparedStatement ps = null;
        ResultSet rs = null;

        try{
            c = ConectorBD.getConnection();

            String sql = "SELECT pf.*, p.nome, p.logradouro, p.cidade,
p.estado,p.telefone, p.email " +
                        "FROM Pessoas_Fisicas pf " +
                        "JOIN Pessoas p ON pf.id_pessoa = p.id_pessoa " +
                        "WHERE pf.id_pessoa = ?";

            ps = c.prepareStatement(sql);
            ps.setInt(1,id);

            rs = ps.executeQuery();

            if(rs.next()){
                pessoa = new PessoaFisica();
                pessoa.setId_pessoa(rs.getInt("id_pessoa"));
                pessoa.setNome(rs.getString("nome"));
                pessoa.setLogradouro(rs.getString("logradouro"));
                pessoa.setCidade(rs.getString("cidade"));
                pessoa.setEstado(rs.getString("estado"));
                pessoa.setTelefone(rs.getString("telefone"));
                pessoa.setEmail(rs.getString("email"));
                pessoa.setCpf(rs.getString("cpf"));
            }

        } catch(SQLException excep){
            excep.printStackTrace();
        } finally {
            ConectorBD.close(c);
            ConectorBD.close(rs);
            ConectorBD.close(ps);
        }

        return pessoa;
    }

    public List<PessoaFisica> getPessoas(){
        List<PessoaFisica> pessoas = new ArrayList<>();
        Connection c = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
    }

```



```

try{
    c = ConectorBD.getConnection();
    String sql = "SELECT pf.*, p.nome, p.logradouro, p.cidade,
p.estado,p.telefone, p.email " +
                "FROM Pessoas_Fisicas pf " +
                "JOIN Pessoas p ON pf.id_pessoa = p.id_pessoa";
    ps = c.prepareStatement(sql);

    rs = ps.executeQuery();

    while(rs.next()){
        PessoaFisica pessoa = new PessoaFisica();
        pessoa.setId_pessoa(rs.getInt("id_pessoa"));
        pessoa.setNome(rs.getString("nome"));
        pessoa.setLogradouro(rs.getString("logradouro"));
        pessoa.setCidade(rs.getString("cidade"));
        pessoa.setEstado(rs.getString("estado"));
        pessoa.setTelefone(rs.getString("telefone"));
        pessoa.setEmail(rs.getString("email"));
        pessoa.setCpf(rs.getString("cpf"));
        pessoas.add(pessoa);
    }
} catch(SQLException excep){
    excep.printStackTrace();
} finally {
    ConectorBD.close(c);
    ConectorBD.close(ps);
    ConectorBD.close(rs);
}
return pessoas;
}

```

```

public boolean incluir(PessoaFisica pessoa){
    Connection c = null;
    PreparedStatement psPessoa = null;
    PreparedStatement psPessoaFisica = null;
    boolean taNoBalde = false;

    try{
        c = ConectorBD.getConnection();

        // inserção dos dados na tabela Pessoas
        String sqlPessoa = "INSERT INTO Pessoas (nome, logradouro,
cidade, estado, telefone, email)" +
                        "VALUES (?, ?, ?, ?, ?, ?)";
        psPessoa = c.prepareStatement(sqlPessoa,
Statement.RETURN_GENERATED_KEYS);
        psPessoa.setString(1, pessoa.getNome());
        psPessoa.setString(2, pessoa.getLogradouro());
        psPessoa.setString(3, pessoa.getCidade());
        psPessoa.setString(4, pessoa.getEstado());

```

```

        psPessoa.setString(5, pessoa.getTelefone());
        psPessoa.setString(6, pessoa.getEmail());
        int linhasAfetadas = psPessoa.executeUpdate();
        if(linhasAfetadas == 0){
            throw new SQLException("Inserção na tabela Pessoas falhou,
nenhuma linha afetada");
        }

        // pegando o id_pessoa gerado pela sequence
        String sqlIdPessoa = "SELECT TOP 1 id_pessoa FROM Pessoas ORDER
BY id_pessoa DESC;";
        PreparedStatement psIdPessoa = c.prepareStatement(sqlIdPessoa);
        ResultSet rsIdPessoa = psIdPessoa.executeQuery();

        int idPessoa = -1;
        if(rsIdPessoa.next()){
            idPessoa = rsIdPessoa.getInt("id_pessoa");
        } else {
            throw new SQLException("Falha ao obter o ID gerado para a
pessoa inserida na tabela Pessoas");
        }

        // inserção de dados na tabela Pessoas_Fisicas
        String sqlPessoaFisica = "INSERT INTO Pessoas_Fisicas
(id_pFisica, id_pessoa, cpf)" +
            "VALUES (?, ?, ?)";
        psPessoaFisica = c.prepareStatement(sqlPessoaFisica);
        psPessoaFisica.setInt(1, idPessoa);
        psPessoaFisica.setInt(2, idPessoa);
        psPessoaFisica.setString(3, pessoa.getCpf());
        psPessoaFisica.executeUpdate();

        taNoBalde = true;
    } catch (SQLException excep){
        excep.printStackTrace();
    } finally {
        ConectorBD.close(c);
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaFisica);
    }
    return taNoBalde;
}

public boolean alterar(PessoaFisica pessoa){
    Connection c = null;
    PreparedStatement psPessoa = null;
    PreparedStatement psPessoaFisica = null;
    boolean taNoBalde = false;

    try{
        c = ConectorBD.getConnection();

        // update da tabela Pessoas

```

```

        String sqlPessoa = "UPDATE Pessoas " +
                            "SET nome = ?, logradouro = ?, cidade = ?,
estado = ?, telefone = ?, email = ? " +
                            "WHERE id_pessoa = ?";
        psPessoa = c.prepareStatement(sqlPessoa);
        psPessoa.setString(1, pessoa.getNome());
        psPessoa.setString(2, pessoa.getLogradouro());
        psPessoa.setString(3, pessoa.getCidade());
        psPessoa.setString(4, pessoa.getEstado());
        psPessoa.setString(5, pessoa.getTelefone());
        psPessoa.setString(6, pessoa.getEmail());
        psPessoa.setInt(7, pessoa.getId_pessoa());
        psPessoa.executeUpdate();

        //update da tabela Pessoas_Fisicas
        String sqlPessoaFisica = "UPDATE Pessoas_Fisicas " +
                                "SET cpf = ? " +
                                "WHERE id_pessoa = ?";
        psPessoaFisica = c.prepareStatement(sqlPessoaFisica);
        psPessoaFisica.setString(1, pessoa.getCpf());
        psPessoaFisica.setInt(2, pessoa.getId_pessoa());
        psPessoaFisica.executeUpdate();

        int linhasAfetadas = psPessoaFisica.executeUpdate();
        taNoBalde = (linhasAfetadas > 0);

    } catch(SQLException excep){
        excep.printStackTrace();
    } finally {
        ConectorBD.close(c);
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaFisica);
    }
    return taNoBalde;
}

public boolean excluir(int id){
    Connection c = null;
    PreparedStatement psPessoa = null;
    PreparedStatement psPessoaFisica = null;
    boolean taNoBalde = false;

    try{
        c = ConectorBD.getConnection();

        // exclusão na tabela Pessoas_Fisicas
        String sqlPessoaFisica = "DELETE FROM Pessoas_Fisicas WHERE
id_pessoa = ?";
        psPessoaFisica = c.prepareStatement(sqlPessoaFisica);
        psPessoaFisica.setInt(1, id);
        psPessoaFisica.executeUpdate();

        // exclusão na tabela Pessoas

```

```

        String sqlPessoa = "DELETE FROM Pessoas WHERE id_pessoa = ?";
        psPessoa = c.prepareStatement(sqlPessoa);
        psPessoa.setInt(1, id);
        psPessoa.executeUpdate();

    } catch (SQLException excep) {
        excep.printStackTrace();
    } finally {
        ConectorBD.close(c);
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaFisica);
    }
    return taNoBalde;
}
}

```

- Classe PessoaJuridicaDAO

```

package cadastro.model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;

import cadastrobd.model.PessoaJuridica;

public class PessoaJuridicaDAO {

    public PessoaJuridica getPessoa(int id){
        PessoaJuridica pessoa = null;
        Connection c = null;
        PreparedStatement ps = null;
        ResultSet rs = null;

        try{
            c = ConectorBD.getConnection();

            String sql = "SELECT pj.*, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email " +
                        "FROM Pessoas_Juridicas pj " +
                        "JOIN Pessoas p ON pj.id_pessoa = p.id_pessoa " +
                        "WHERE pj.id_pessoa = ?";
            ps = c.prepareStatement(sql);
            ps.setInt(1, id);

            rs = ps.executeQuery();

```

```

        if(rs.next()){
            pessoa = new PessoaJuridica();
            pessoa.setId_pessoa(rs.getInt("id_pessoa"));
            pessoa.setNome(rs.getString("nome"));
            pessoa.setLogradouro(rs.getString("logradouro"));
            pessoa.setCidade(rs.getString("cidade"));
            pessoa.setEstado(rs.getString("estado"));
            pessoa.setTelefone(rs.getString("telefone"));
            pessoa.setEmail(rs.getString("email"));
            pessoa.setCnpj(rs.getString("cnpj"));

        }

    } catch(SQLException excep){
        excep.printStackTrace();
    }
    return pessoa;
}

public List<PessoaJuridica> getPessoas(){
    List<PessoaJuridica> pessoas = new ArrayList<>();
    Connection c = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try{
        c = ConectorBD.getConnection();

        String sql = "SELECT pj.*, p.nome, p.logradouro, p.cidade,
p.estado, p.telefone, p.email " +
                    "FROM Pessoas_Juridicas pj " +
                    "JOIN Pessoas p ON pj.id_pessoa = p.id_pessoa";
        ps = c.prepareStatement(sql);
        rs = ps.executeQuery();

        while (rs.next()){
            PessoaJuridica pessoa = new PessoaJuridica();
            pessoa.setId_pessoa(rs.getInt("id_pessoa"));
            pessoa.setNome(rs.getString("nome"));
            pessoa.setLogradouro(rs.getString("logradouro"));
            pessoa.setCidade(rs.getString("cidade"));
            pessoa.setEstado(rs.getString("estado"));
            pessoa.setTelefone(rs.getString("telefone"));
            pessoa.setEmail(rs.getString("email"));
            pessoa.setCnpj(rs.getString("cnpj"));

            pessoas.add(pessoa);
        }

    } catch(SQLException excep){
        excep.printStackTrace();
    } finally {
        ConectorBD.close(c);
    }
}

```

```

        ConectorBD.close(ps);
        ConectorBD.close(rs);
    }
    return pessoas;
}

public boolean incluir(PessoaJuridica pessoa){
    Connection c = null;
    PreparedStatement psPessoa = null;
    PreparedStatement psPessoaJuridica = null;
    boolean taNoBalde = false;

    try{
        c = ConectorBD.getConnection();

        String sqlPessoa = "INSERT INTO Pessoas(nome, logradouro, cidade,
estado, telefone, email) " +
            "VALUES (?, ?, ?, ?, ?, ?)";
        psPessoa = c.prepareStatement(sqlPessoa,
Statement.RETURN_GENERATED_KEYS);
        psPessoa.setString(1, pessoa.getNome());
        psPessoa.setString(2, pessoa.getLogradouro());
        psPessoa.setString(3, pessoa.getCidade());
        psPessoa.setString(4, pessoa.getEstado());
        psPessoa.setString(5, pessoa.getTelefone());
        psPessoa.setString(6, pessoa.getEmail());
        int linhasAfetadas = psPessoa.executeUpdate();
        if(linhasAfetadas == 0){
            throw new SQLException("Inserção na tabela Pessoas falhou,
nenhuma linha afetada");
        }

        String sqlIdPessoa = "SELECT TOP 1 id_pessoa FROM Pessoas ORDER
BY id_pessoa DESC;";
        PreparedStatement psIdPessoa = c.prepareStatement(sqlIdPessoa);
        ResultSet rsIdPessoa = psIdPessoa.executeQuery();

        int idPessoa = -1;
        if(rsIdPessoa.next()){
            idPessoa = rsIdPessoa.getInt("id_pessoa");
        } else {
            throw new SQLException("Falha ao obter o ID gerado para a
pessoa inserida na tabela Pessoas");
        }

        String sqlPessoaFisica = "INSERT INTO Pessoas_Juridicas
(id_pJuridica, id_pessoa, cnpj) " +
            "VALUES (?, ?, ?)";
        psPessoaJuridica = c.prepareStatement(sqlPessoaFisica);
        psPessoaJuridica.setInt(1, idPessoa);
        psPessoaJuridica.setInt(2, idPessoa);
        psPessoaJuridica.setString(3, pessoa.getCnpj());
    }
}

```

```

        psPessoaJuridica.executeUpdate();

        taNoBalde = true;
    } catch(SQLException excep){
        excep.printStackTrace();
    } finally {
        ConectorBD.close(c);
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaJuridica);
    }
    return taNoBalde;
}

public boolean alterar(PessoaJuridica pessoa){
    Connection c = null;
    PreparedStatement psPessoa = null;
    PreparedStatement psPessoaJuridica = null;
    boolean taNoBalde = false;

    try{
        c = ConectorBD.getConnection();

        String sqlPessoa = "UPDATE Pessoas " +
            "SET nome = ?, logradouro = ?, cidade = ?,
estado = ?, telefone = ?, email = ? " +
            "WHERE id_pessoa = ?";
        psPessoa = c.prepareStatement(sqlPessoa);
        psPessoa.setString(1, pessoa.getNome());
        psPessoa.setString(2, pessoa.getLogradouro());
        psPessoa.setString(3, pessoa.getCidade());
        psPessoa.setString(4, pessoa.getEstado());
        psPessoa.setString(5, pessoa.getTelefone());
        psPessoa.setString(6, pessoa.getEmail());
        psPessoa.setInt(7, pessoa.getId_pessoa());
        psPessoa.executeUpdate();

        String sqlPessoaJuridica = "UPDATE Pessoas_Juridicas " +
            "SET cnpj = ? " +
            "WHERE id_pessoa = ?";
        psPessoaJuridica = c.prepareStatement(sqlPessoaJuridica);
        psPessoaJuridica.setString(1,pessoa.getCnpj());
        psPessoaJuridica.setInt(2,pessoa.getId_pessoa());
        psPessoaJuridica.executeUpdate();

        int linhasAfetadas = psPessoaJuridica.executeUpdate();
        taNoBalde = (linhasAfetadas > 0);

    } catch(SQLException excep){
        excep.printStackTrace();
    } finally {
        ConectorBD.close(c);
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaJuridica);
    }
}

```

```

    }
    return taNoBalde;
}

public boolean excluir(int id){
    Connection c = null;
    PreparedStatement psPessoa = null;
    PreparedStatement psPessoaJuridica = null;
    boolean taNoBalde = false;

    try {
        c = ConectorBD.getConnection();

        String sqlPessoaJuridica = "DELETE FROM Pessoas_Juridicas WHERE
id_pessoa = ?";
        psPessoaJuridica = c.prepareStatement(sqlPessoaJuridica);
        psPessoaJuridica.setInt(1, id);
        psPessoaJuridica.executeUpdate();

        String sqlPessoa = "DELETE FROM Pessoas WHERE id_pessoa = ?";
        psPessoa = c.prepareStatement(sqlPessoa);
        psPessoa.setInt(1, id);
        psPessoa.executeUpdate();

    } catch (SQLException excep){
        excep.printStackTrace();
    } finally {
        ConectorBD.close(c);
        ConectorBD.close(psPessoa);
        ConectorBD.close(psPessoaJuridica);
    }
    return taNoBalde;
}
}

```

Classe principal de teste: **CadastroBDTeste**

```

/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastrobd;

import cadastrobd.model.PessoaFisica;
import cadastro.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastro.model.PessoaJuridicaDAO;
import java.sql.SQLException;
import java.util.List;

```



```

/**
 *
 * @author anaqu
 */
public class CadastroBDTeste {

    public void testeInserir(){
        //criando uma pessoa fisica e testando o incluir()
        PessoaFisica pessoaFisica = new PessoaFisica();
        pessoaFisica.setNome("Rachel Green");
        pessoaFisica.setLogradouro("Central Park");
        pessoaFisica.setCidade("New York City");
        pessoaFisica.setEstado("NY");
        pessoaFisica.setTelefone("+1 12345678");
        pessoaFisica.setEmail("rachel@friends.com");
        pessoaFisica.setCpf("12345678910");

        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

        boolean deuCerto = pessoaFisicaDAO.incluir(pessoaFisica);
        if(deuCerto){
            System.out.println("DEU CERTOO");
        } else {
            System.out.println("tururu NÃO DEU CERTO");
        }
        // criando uma pessoa jurídica e testando o incluir()
        PessoaJuridica pessoaJuridica = new PessoaJuridica();
        pessoaJuridica.setNome("Friends LTDA");
        pessoaJuridica.setLogradouro("Central PERk");
        pessoaJuridica.setCidade("NYC");
        pessoaJuridica.setEstado("NY");
        pessoaJuridica.setTelefone("+1 77777777");
        pessoaJuridica.setEmail("friends@friends.com");
        pessoaJuridica.setCnpj("7700077700017");

        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
        boolean deuCerto2 = pessoaJuridicaDAO.incluir(pessoaJuridica);
        if(deuCerto2){
            System.out.println("DEU CERTOO");
        } else {
            System.out.println("tururu NÃO DEU CERTO");
        }
    }

    public void testeAlterar() throws SQLException{
        // testando o alterar()
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        PessoaFisica pf = new PessoaFisica();

        pf.setId_pessoa(11);
        pf = pessoaFisicaDAO.getPessoa(pf.getId_pessoa());
    }
}

```

```

        if(pf == null){
            System.out.println("Pessoa não encontrada");
        } else {
            pf.setNome("Chandler Bing");
            pf.setEmail("chandler@friends.com");
        }

        boolean deuCerto = pessoaFisicaDAO.alterar(pf);
        if(deuCerto){
            System.out.println("DEU CERTOO");
        } else {
            System.out.println("tururu NÃO DEU CERTO");
        }

        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
        PessoaJuridica pj = new PessoaJuridica();

        pj.setId_pessoa(19);
        pj = pessoaJuridicaDAO.getPessoa(pj.getId_pessoa());

        if(pj == null){
            System.out.println("Empresa não encontrada");
        } else {
            pj.setNome("Central Perk Coffee");
            pj.setEmail("coffee@friends.com");
        }

        boolean deuCerto2 = pessoaJuridicaDAO.alterar(pj);
        if(deuCerto2){
            System.out.println("DEU CERTOO");
        } else {
            System.out.println("tururu NÃO DEU CERTO");
        }
    }

    public void testeExibirPessoas() throws SQLException{
        PessoaFisicaDAO pFdao = new PessoaFisicaDAO();
        List<PessoaFisica> pessoasFisicas = pFdao.getPessoas();

        PessoaJuridicaDAO pJdao = new PessoaJuridicaDAO();
        List<PessoaJuridica> pessoasJuridicas = pJdao.getPessoas();

        if(pessoasFisicas != null && !pessoasFisicas.isEmpty()){
            System.out.println("Lista de todas as pessoas físicas no banco de
dados:");

            System.out.println("=====");
            for(PessoaFisica pessoa : pessoasFisicas){
                pessoa.exibir();
                System.out.println("-----");
            }
        }
    }

```

```

    }
    if(pessoasJuridicas != null && !pessoasJuridicas.isEmpty()){
        System.out.println("Lista de todas as pessoas jurídicas no banco
de dados:");

System.out.println("=====");
        for(PessoaJuridica pessoa : pessoasJuridicas){
            pessoa.exibir();
            System.out.println("-----");
        }
    }
}

public void testeExcluir(String type,int id){

    switch(type.toUpperCase()){
        case "F":
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
            pessoaFisicaDAO.excluir(id);
            break;
        case "J":
            PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO();
            pessoaJuridicaDAO.excluir(id);
            break;
        default:
            System.out.println("Tipo inválido");
    }
}

public static void main(String[] args) throws SQLException{

    CadastroBDTeste cadastro = new CadastroBDTeste();

    //TESTE PARA INSERIR DADOS NA BD
    cadastro.testeInserir();

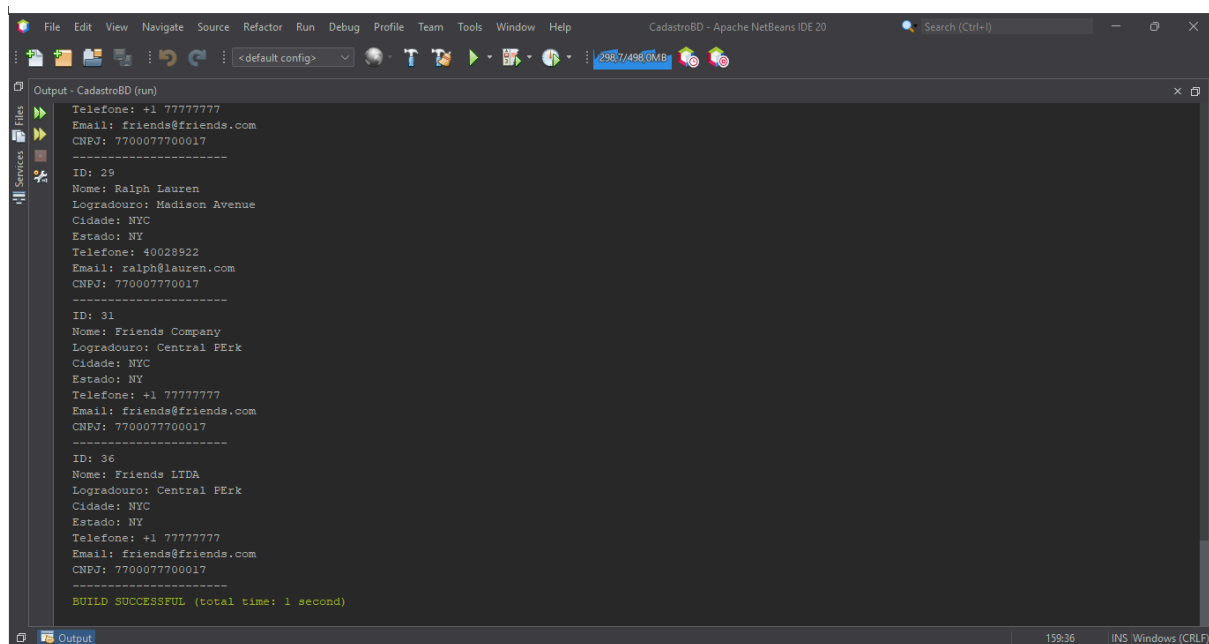
    // TESTE PARA ALTERAR DADOS NA BD
    cadastro.testeAlterar();

    // TESTE DE CONSULTAR TODAS AS PESSOAS DO BD
    cadastro.testeExibirPessoas();

    //TESTE PARA EXCLUIR DADOS NA BD
    cadastro.testeExcluir("f",6);
}
}

```

Saída do sistema ao executar a classe CadastroBDTeste:



```
Output - CadastroBD (run)
Telefone: +1 77777777
Email: friends@friends.com
CNPJ: 7700077700017
-----
ID: 29
Nome: Ralph Lauren
Logradouro: Madison Avenue
Cidade: NYC
Estado: NY
Telefone: 40028922
Email: ralph@lauren.com
CNPJ: 7700077700017
-----
ID: 31
Nome: Friends Company
Logradouro: Central PErk
Cidade: NYC
Estado: NY
Telefone: +1 77777777
Email: friends@friends.com
CNPJ: 7700077700017
-----
ID: 36
Nome: Friends LTDA
Logradouro: Central PErk
Cidade: NYC
Estado: NY
Telefone: +1 77777777
Email: friends@friends.com
CNPJ: 7700077700017
-----
BUILD SUCCESSFUL (total time: 1 second)
```

Análises e Conclusões

- a. O middleware funciona como uma ponte entre aplicações e banco de dados, fornecendo uma interface que facilita a comunicação entre aplicação e banco de dados, otimizando o desenvolvimento, a escalabilidade e a segurança das aplicações.
- b. O `Statement` é mais simples, bom para consultas simples e rápidas, como ler ou inserir dados. Já o `PreparedStatement` tem a característica de separar o código SQL dos dados, protegendo o sistema contra SQL injection, o que faz dele, em comparação ao `Statement`, mais seguro.
Resumindo, o `PreparedStatement` é mais seguro e tem a capacidade de fazer consultas mais complexas, enquanto o `Statement` é menos seguro e usado para consultas mais simples.
- c. O uso do padrão DAO proporciona benefícios como modularidade, reutilização do código, facilidade de testes, flexibilidade e clareza do código. Logo, ao proporcionar tudo isso, o padrão deixa a manutenção do código mais prática.
- d. Quando lidamos com um modelo estritamente relacional a herança pode ser refletida de duas formas: com tabela única, a qual as classes filhas compartilham a mesma tabela tendo um coluna para identificar o tipo de cada; e com tabela por subtipo, onde a classe filha tem a sua própria tabela no banco de dados.

2º Procedimento | Alimentando a Base

Método main da classe principal do projeto:

```
/*
 * Click
 nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
 change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit
 this template
 */
package cadastrobd;

import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;

/**
 *
 * @author anaqu
 */
public class CadastroBD {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws SQLException{
        Scanner sc = new Scanner(System.in);
        PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
        PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
        int opcao;
        String tipoPessoa, resposta;
        boolean deuCerto;

        do {
            System.out.println("=====");
            System.out.println("1 - Incluir Pessoa");
            System.out.println("2 - Alterar Pessoa");
            System.out.println("3 - Excluir Pessoa");
            System.out.println("4 - Buscar pelo ID");
            System.out.println("5 - Exibir Todos");
            System.out.println("0 - Finalizar Programa");
            System.out.println("=====");
            opcao = Integer.parseInt(sc.nextLine());
            switch(opcao) {

                case 1:
                    System.out.println("F - Pessoas Físicas | J - Pessoas
Jurídicas");
                    tipoPessoa = sc.nextLine();

```

```

if(tipoPessoa.equalsIgnoreCase("F")){
    PessoaFisica pf = new PessoaFisica();

    System.out.println("Digite o nome da pessoa:");
    String nome = sc.nextLine();
    pf.setNome(nome);
    System.out.println("Digite o logradouro da pessoa:");
    String logradouro = sc.nextLine();
    pf.setLogradouro(logradouro);
    System.out.println("Digite a cidade da pessoa:");
    String cidade = sc.nextLine();
    pf.setCidade(cidade);
    System.out.println("Digite o estado da pessoa:");
    String estado = sc.nextLine();
    pf.setEstado(estado);
    System.out.println("Digite o telefone da pessoa: (11
dígitos)");

    String telefone = sc.nextLine();
    pf.setTelefone(telefone);
    System.out.println("Digite o email da pessoa:");
    String email = sc.nextLine();
    pf.setEmail(email);
    System.out.println("Digite o CPF da pessoa: (Sem
ponto e traço)");

    String cpf = sc.nextLine();
    pf.setCpf(cpf);

    deuCerto = pfDAO.incluir(pf);
    if(deuCerto){
        System.out.println("Pessoa incluída com
sucesso!");
    } else {
        System.out.println("Falha na inclusão de
dados.");
    }
} else if (tipoPessoa.equalsIgnoreCase("J")){
    PessoaJuridica pj = new PessoaJuridica();

    System.out.println("Digite o nome da empresa:");
    String nome = sc.nextLine();
    pj.setNome(nome);
    System.out.println("Digite o logradouro da
empresa:");

    String logradouro = sc.nextLine();
    pj.setLogradouro(logradouro);
    System.out.println("Digite a cidade da empresa:");
    String cidade = sc.nextLine();
    pj.setCidade(cidade);
    System.out.println("Digite o estado da empresa:");
    String estado = sc.nextLine();
    pj.setEstado(estado);
    System.out.println("Digite o telefone da empresa:");

```

```

        String telefone = sc.nextLine();
        pj.setTelefone(telefone);
        System.out.println("Digite o email da empresa:");
        String email = sc.nextLine();
        pj.setEmail(email);
        System.out.println("Digite o CNPJ da empresa:");
        String cnpj = sc.nextLine();
        pj.setCnpj(cnpj);

        deuCerto = pjDAO.incluir(pj);
        if(deuCerto){
            System.out.println("Empresa incluída com
sucesso!");

        } else {
            System.out.println("Falha na inclusão de
dados.");

        }
    } else {
        System.out.println("Opção inválida");
    }
    break;
case 2:
    System.out.println("F - Pessoas Físicas | J - Pessoas
Jurídicas");

    tipoPessoa = sc.nextLine();
    if(tipoPessoa.equalsIgnoreCase("F")){
        PessoaFisica pf = new PessoaFisica();

        System.out.println("Digite o ID da pessoa que você
quer alterar:");

        int id = Integer.parseInt(sc.nextLine());
        pf = pfDAO.getPessoa(id);

        if(pf == null){
            System.out.println("ID inexistente no banco de
dados.");

        } else {
            pf.exibir();

            System.out.println("Deseja mudar o nome? (S/N)");
            resposta = sc.nextLine();
            if(resposta.equalsIgnoreCase("S")){
                System.out.println("Entre com o novo nome:");
                String nome = sc.nextLine();
                pf.setNome(nome);
            }
            System.out.println("Deseja mudar o logradouro?
(S/N)");

            resposta = sc.nextLine();
            if(resposta.equalsIgnoreCase("S")){
                System.out.println("Entre com o novo
logradouro:");

                String logradouro = sc.nextLine();

```

```

        pf.setLogradouro(logradouro);
    }
    System.out.println("Deseja mudar a cidade?
(S/N)");

    resposta = sc.nextLine();
    if(resposta.equalsIgnoreCase("S")){
        System.out.println("Entre com a nova
cidade:");

        String cidade = sc.nextLine();
        pf.setCidade(cidade);
    }
    System.out.println("Deseja mudar o estado?
(S/N)");

    resposta = sc.nextLine();
    if(resposta.equalsIgnoreCase("S")){
        System.out.println("Entre com o novo
estado:");

        String estado = sc.nextLine();
        pf.setEstado(estado);
    }
    System.out.println("Deseja mudar o telefone?
(S/N)");

    resposta = sc.nextLine();
    if(resposta.equalsIgnoreCase("S")){
        System.out.println("Entre com o novo
telefone:");

        String telefone = sc.nextLine();
        pf.setTelefone(telefone);
    }
    System.out.println("Deseja mudar o email?
(S/N)");

    resposta = sc.nextLine();
    if(resposta.equalsIgnoreCase("S")){
        System.out.println("Entre com o novo
email:");

        String email = sc.nextLine();
        pf.setEmail(email);
    }
    System.out.println("Deseja mudar o CPF? (S/N)");
    resposta = sc.nextLine();
    if(resposta.equalsIgnoreCase("S")){
        System.out.println("Entre com o novo CPF:");
        String cpf = sc.nextLine();
        pf.setCpf(cpf);
    }
    deuCerto = pfDAO.alterar(pf);
    if(deuCerto){
        System.out.println("Pessoa alterada com
sucesso!");
    } else {
        System.out.println("Falha na alteração de
dados.");
    }
}

```



```

    }
    } else if (tipoPessoa.equalsIgnoreCase("J")) {
        PessoaJuridica pj = new PessoaJuridica();

        System.out.println("Digite o ID da empresa que você
quer alterar:");

        int id = Integer.parseInt(sc.nextLine());
        pj = pjDAO.getPessoa(id);
        if(pj == null){
            System.out.println("ID inexistente no banco de
dados.");

        } else {
            pj.exibir();

            System.out.println("Deseja mudar o nome? (S/N)");
            resposta = sc.nextLine();
            if(resposta.equalsIgnoreCase("S")){
                System.out.println("Entre com o novo nome:");
                String nome = sc.nextLine();
                pj.setNome(nome);
            }
            System.out.println("Deseja mudar o logradouro?
(S/N)");

            resposta = sc.nextLine();
            if(resposta.equalsIgnoreCase("S")){
                System.out.println("Entre com o novo
logradouro:");

                String logradouro = sc.nextLine();
                pj.setLogradouro(logradouro);
            }
            System.out.println("Deseja mudar a cidade?
(S/N)");

            resposta = sc.nextLine();
            if(resposta.equalsIgnoreCase("S")){
                System.out.println("Entre com a nova
cidade:");

                String cidade = sc.nextLine();
                pj.setCidade(cidade);
            }
            System.out.println("Deseja mudar o estado?
(S/N)");

            resposta = sc.nextLine();
            if(resposta.equalsIgnoreCase("S")){
                System.out.println("Entre com o novo
estado:");

                String estado = sc.nextLine();
                pj.setEstado(estado);
            }
            System.out.println("Deseja mudar o telefone?
(S/N)");

            resposta = sc.nextLine();
            if(resposta.equalsIgnoreCase("S")){

```

```

        System.out.println("Entre com o novo
telefone:");

        String telefone = sc.nextLine();
        pj.setTelefone(telefone);
    }
    System.out.println("Deseja mudar o email?
(S/N)");

    resposta = sc.nextLine();
    if(resposta.equalsIgnoreCase("S")){
        System.out.println("Entre com o novo
email:");

        String email = sc.nextLine();
        pj.setEmail(email);
    }
    System.out.println("Deseja mudar o CNPJ? (S/N)");
    resposta = sc.nextLine();
    if(resposta.equalsIgnoreCase("S")){
        System.out.println("Entre com o novo CNPJ:");
        String cnpj = sc.nextLine();
        pj.setCnpj(cnpj);
    }
    deuCerto = pjDAO.alterar(pj);
    if(deuCerto){
        System.out.println("Pessoa alterada com
sucesso!");
    } else {
        System.out.println("Falha na alteração de
dados.");
    }
} else {
    System.out.println("Opção inválida");
}
break;
case 3:
    System.out.println("F - Pessoas Físicas | J - Pessoas
Jurídicas");

    tipoPessoa = sc.nextLine();
    if(tipoPessoa.equalsIgnoreCase("F")){
        PessoaFisica pf = new PessoaFisica();

        System.out.println("Digite o ID da pessoa que você
quer excluir:");

        int id = Integer.parseInt(sc.nextLine());
        pf = pfDAO.getPessoa(id);

        if(pf == null){
            System.out.println("ID inexistente no bando de
dados.");
        } else {
            pf.exibir();

```

```

        System.out.println("Certeza que deseja excluir
essa pessoa do banco de dados? (S/N)");
        resposta = sc.nextLine();
        if(resposta.equalsIgnoreCase("S")){
            pfDAO.excluir(id);
            System.out.println("Pessoa excluída do banco
de dados.");
        } else {
            System.out.println("Exclusão cancelada.");
        }
    }
} else if (tipoPessoa.equalsIgnoreCase("J")){
    PessoaJuridica pj = new PessoaJuridica();

    System.out.println("Digite o ID da empresa que você
quer excluir:");

    int id = Integer.parseInt(sc.nextLine());
    pj = pjDAO.getPessoa(id);

    if(pj == null){
        System.out.println("ID inexistente no banco de
dados.");
    } else {
        pj.exibir();

        System.out.println("Certeza que deseja excluir
essa empresa do banco de dados? (S/N)");
        resposta = sc.nextLine();
        if(resposta.equalsIgnoreCase("S")){
            pjDAO.excluir(id);
            System.out.println("Empresa excluída do banco
de dados.");
        } else {
            System.out.println("Exclusão cancelada.");
        }
    }
} else {
    System.out.println("Opção inválida");
}
break;
case 4:
    System.out.println("F - Pessoas Físicas | J - Pessoas
Jurídicas");

    tipoPessoa = sc.nextLine();
    if(tipoPessoa.equalsIgnoreCase("F")){
        PessoaFisica pf = new PessoaFisica();

        System.out.println("Digite o ID da pessoa que você
quer ver:");

        int id = Integer.parseInt(sc.nextLine());
        pf = pfDAO.getPessoa(id);
        pf.exibir();
    }
}

```

```

    } else if (tipoPessoa.equalsIgnoreCase("J")){
        PessoaJuridica pj = new PessoaJuridica();

        System.out.println("Digite o ID da empresa que você
quer ver:");

        int id = Integer.parseInt(sc.nextLine());
        pj = pjDAO.getPessoa(id);
        pj.exibir();
    } else {
        System.out.println("Opção inválida");
    }
    break;
case 5:
    System.out.println("F - Pessoas Físicas | J - Pessoas
Jurídicas");

    tipoPessoa = sc.nextLine();
    if(tipoPessoa.equalsIgnoreCase("F")){
        List<PessoaFisica> listaPf = pfDAO.getPessoas();

        if(listaPf != null && !listaPf.isEmpty()){
            System.out.println("Lista de todas as pessoas físicas
no banco de dados:");

            System.out.println("=====");
            for(PessoaFisica pessoa : listaPf){
                pessoa.exibir();
                System.out.println("-----");
            }

        } else if (tipoPessoa.equalsIgnoreCase("J")){
            List<PessoaJuridica> listaPj = pjDAO.getPessoas();

            if(listaPj != null && !listaPj.isEmpty()){
                System.out.println("Lista de todas as empresas no
banco de dados:");

                System.out.println("=====");
                for(PessoaJuridica pessoa : listaPj){
                    pessoa.exibir();
                    System.out.println("-----");
                }

            }

        } else {
            System.out.println("Opção inválida");
        }
        break;
case 0:
    System.out.println("Finalizando o programa...");
    break;
default:
    System.out.println("Opção inválida");
    break;
}

```

```

        } while (opcao != 0);
        sc.close();
    }
}

```

Saída do sistema ao executar a classe main

1. Incluir Pessoa

```

run:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
1
F - Pessoas Fisicas | J - Pessoas Juridicas
F
Digite o nome da pessoa:
Phoebe Buffay
Digite o logradouro da pessoa:
Central Park
Digite a cidade da pessoa:
NYC
Digite o estado da pessoa:
NY
Digite o telefone da pessoa: (11 digitos)
12345678910
Digite o email da pessoa:
phoebe@buffay.com
Digite o CPF da pessoa: (Sem ponto e traço)
12345678952
Pessoa incluída com sucesso!
=====

```

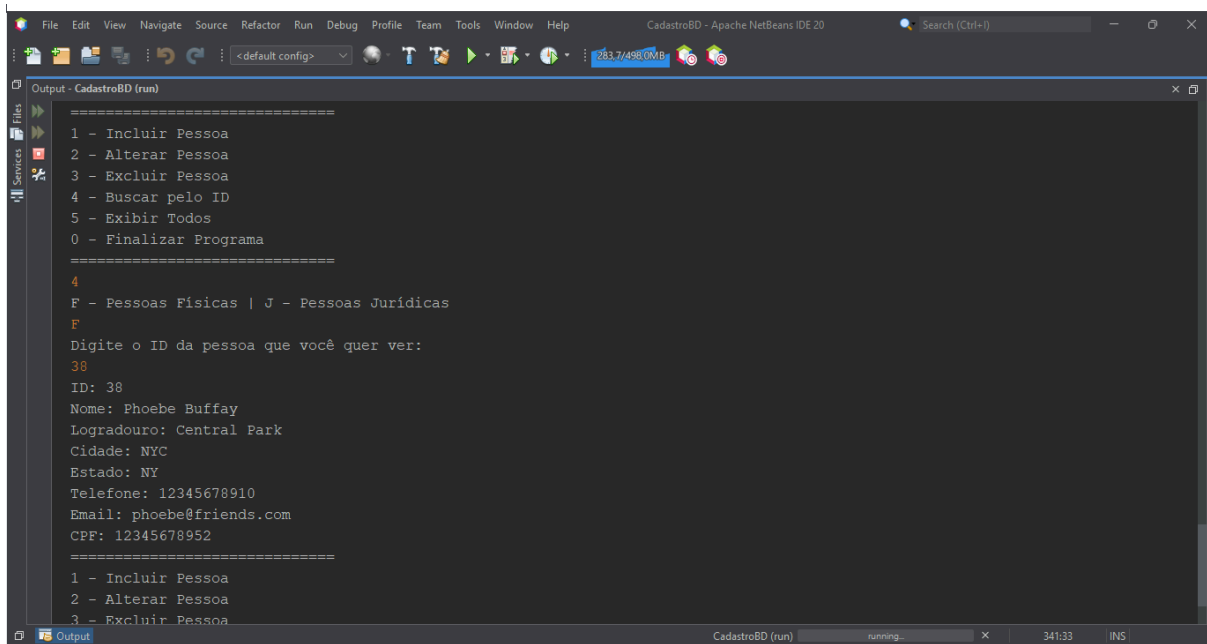
2. Alterar Pessoa

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help CadastroBD - Apache NetBeans IDE 20 Search (Ctrl+I)
Output - CadastroBD (run)
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
2
F - Pessoas Físicas | J - Pessoas Jurídicas
F
Digite o ID da pessoa que você quer alterar:
38
ID: 38
Nome: Phoebe Buffay
Logradouro: Central Park
Cidade: NYC
Estado: NY
Telefone: 12345678910
Email: phoebe@buffay.com
CPF: 12345678952
Deseja mudar o nome? (S/N)
n
Deseja mudar o logradouro? (S/N)
n
Deseja mudar a cidade? (S/N)
n
Deseja mudar o estado? (S/N)
n
Deseja mudar o telefone? (S/N)
n
Deseja mudar o email? (S/N)
s
Entre com o novo email:
phoebe@friends.com
Deseja mudar o CPF? (S/N)
n
Pessoa alterada com sucesso!
```

3. Excluir Pessoa

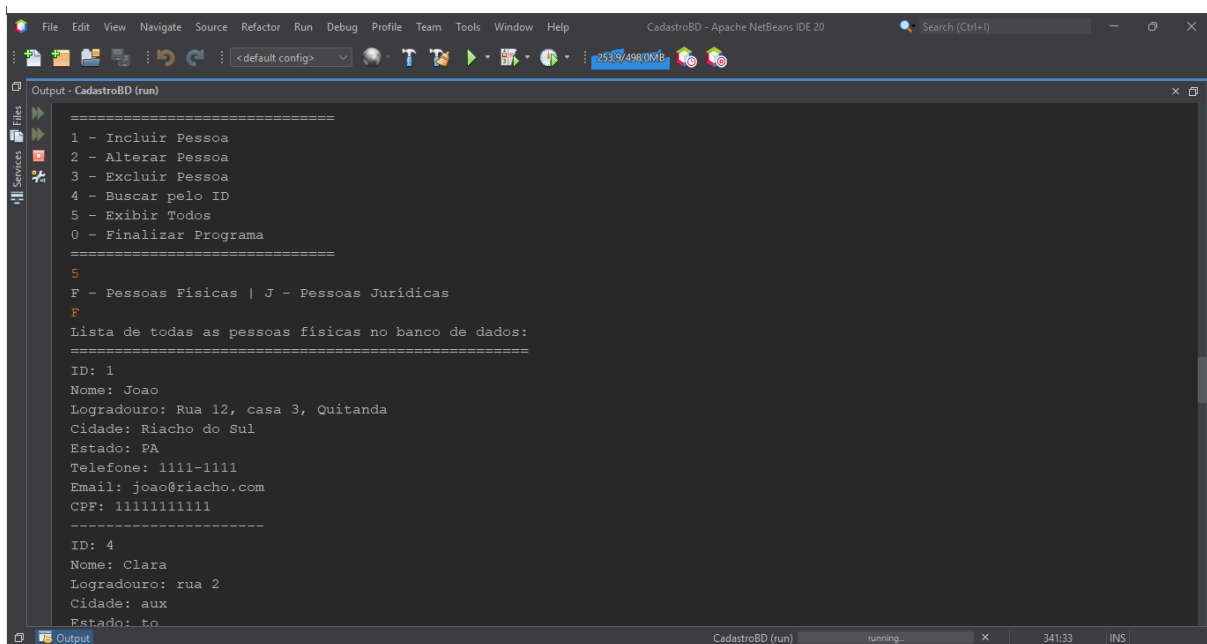
```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help CadastroBD - Apache NetBeans IDE 20 Search (Ctrl+I)
Output - CadastroBD (run)
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
3
F - Pessoas Físicas | J - Pessoas Jurídicas
J
Digite o ID da empresa que você quer excluir:
37
ID: 37
Nome: 0
Logradouro: 0
Cidade: 0
Estado: 0
Telefone: 000
Email: 0
CNPJ: 0
Certeza que dejesa excluir essa empresa do banco de dados? (S/N)
S
Empresa excluída do banco de dados.
=====
```

4. Buscar pelo ID



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help CadastroBD - Apache NetBeans IDE 20 Search (Ctrl+I)
Output - CadastroBD (run)
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
4
F - Pessoas Físicas | J - Pessoas Jurídicas
F
Digite o ID da pessoa que você quer ver:
38
ID: 38
Nome: Phoebe Buffay
Logradouro: Central Park
Cidade: NYC
Estado: NY
Telefone: 12345678910
Email: phoebe@friends.com
CPF: 12345678952
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
=====
CadastroBD (run) running... 341:33 INS
```

5. Exibir Todos



```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help CadastroBD - Apache NetBeans IDE 20 Search (Ctrl+I)
Output - CadastroBD (run)
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
0 - Finalizar Programa
=====
5
F - Pessoas Físicas | J - Pessoas Jurídicas
F
Lista de todas as pessoas físicas no banco de dados:
=====
ID: 1
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 11111111111
-----
ID: 4
Nome: Clara
Logradouro: rua 2
Cidade: aux
Estado: to
-----
CadastroBD (run) running... 341:33 INS
```

Pessoa Física

```
=====  
1 - Incluir Pessoa  
2 - Alterar Pessoa  
3 - Excluir Pessoa  
4 - Buscar pelo ID  
5 - Exibir Todos  
0 - Finalizar Programa  
=====
```

5

F - Pessoas Físicas | J - Pessoas Jurídicas

J

Lista de todas as empresas no banco de dados:

=====

ID: 2
Nome: JCC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jcc@riacho.com
CNPJ: 22222222222222

ID: 19
Nome: Central Perk Coffee
Logradouro: Central PERk
Cidade: NYC
Estado: NY

Pessoa Jurídica

6. Finalizar Programa

```
Cidade: NYC  
Estado: NY  
Telefone: +1 77777777  
Email: friends@friends.com  
CNPJ: 7700077700017  
-----  
ID: 36  
Nome: Friends LTDA  
Logradouro: Central PERk  
Cidade: NYC  
Estado: NY  
Telefone: +1 77777777  
Email: friends@friends.com  
CNPJ: 7700077700017  
-----  
1 - Incluir Pessoa  
2 - Alterar Pessoa  
3 - Excluir Pessoa  
4 - Buscar pelo ID  
5 - Exibir Todos  
0 - Finalizar Programa  
=====
```

0

Finalizando o programa...

BUILD SUCCESSFUL (total time: 8 minutes 34 seconds)

Análise e conclusões:

- a. Diferenças entre persistência em arquivos e persistência em banco de dados:
 - i. Estrutura de armazenamento - a persistência em arquivos armazena os dados em arquivos no sistema operacional, podendo ser de vários tipo; já a persistência em dados armazena os dados em tabelas estruturadas dentro do banco de dados.
 - ii. Consulta e manipulação de dados - a persistência em arquivos geralmente requer operações de leitura e gravações de arquivos diretamente no sistema de arquivos para fazer a manipulação de

dados; já a persistência em banco de dados tem as operações CRUD que são realizadas por meio de consultas SQL.

- iii. Concorrência e transações - Na persistência em dados é mais difícil lidar com concorrência e garantir a integridade dos dados; enquanto na persistência em banco de dados, os bancos de dados oferecem recursos para controlar a concorrência garantindo a consistência dos dados.
 - iv. Escalabilidade - a persistência em arquivos é muito limitada comparada à persistência em banco de dados, onde os banco de dados são projetados para escalabilidade e podem lidar com grandes volumes de dados e cargas de trabalho.
 - v. Segurança e controle de acesso - a segurança da persistência em arquivo depende da permissão do sistema de arquivos, enquanto a persistência em banco de dados oferece recursos avançados de segurança, como autenticação, autorização, criptografia e auditoria.
- b. O uso de operador lambda simplificou a impressão dos valores contidos nas entidades. Um exemplo do uso de lambda está na linha 336 do método main, quando é pedido para imprimir uma lista de todas as pessoas físicas no banco de dados.
 - c. Para que o método seja reconhecido como pertencente à classe em si e não instâncias individuais, assim eliminando a necessidade de instanciar um objeto apenas para acessar o método.